Escuela
Politécnica
Superior

Multilingual Detection of Hate Speech Against Women and Immigrants in Twitter

Bachelor's Degree in Computer Engineering

Bachelor's Thesis

Author:
Carlos Perelló Camacho

Tutors:
Jose Garcia Rodriguez
David Tomás Díaz
Alberto Garcia Garcia

June 2019

Universitat d'Alacant
Universidad de Alicante

Multilingual Detection of Hate Speech against Women and Immigrants in Twitter

Author

Carlos Perelló Camacho

Tutors

Jose Garcia Rodriguez

Department of Computer Technology (DTIC)

David Tomás Díaz

Department of Software and Computing Systems (DLSI)

Alberto Garcia Garcia

Department of Computer Technology (DTIC)

Degree in Computer Engineering

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

Alicante, June, 2019

Copy from one, it's plagiarism; copy from two, it's research.

Wilson Mizner

Acknowledgements

I would like to express my sincere gratitude to my advisors Jose, David and Albert. Thank you for your knowledge, your commitment and your availability.

My sincere thanks also goes to my cousins Jose Camacho and Miguel Camacho for your continuous help during the development of this project and the Hate Crime National Office for their support.

Last but not least, I would like to thank my family and my colleagues, this project would not have been possible without you.

Abstract

Due to the massive rise of users in social media, the presence of verbal abuse, hate speech and bully-attitudes has increased over the years. Especially on Twitter, users find a way to anonymously harass and offend other individuals or collectives, and not enough work is done to stop them.

This project describes the implementation of our hate speech detection system against women and immigrants with the aim of serving to reduce hatred in social networks in the future and participating in the SemEval-2019 Task 5 challenge.

SemEval-2019 Task 5 consists of detecting hate speech against women and immigrants in Twitter, both in English and Spanish. This work proposes a strong baseline for hate speech detection by means of a traditional Machine Learning techniques. Our system is mainly based on the use of n-grams, sentiment analysis and word embeddings together.

In the challenge, given the text of a tweet, one of the tasks consists of identifying hate speech against women and immigrants. Our system obtained the second highest accuracy in Task A in Spanish, surpassing more complex systems based on neural networks of a total of 40 participants.

Resumen

Debido al aumento masivo de usuarios en redes sociales, la presencia de abuso verbal, el discurso de odio y las actitudes violentas han aumentado en los últimos años. Especialmente en Twitter, los usuarios encuentran una manera de atacar y ofender anónimamente a otros individuos y collectivos, y no se trabaja lo suficiente para detenerlos.

Este proyecto describe la implementación de nuestro sistema de detección de discurso de odio contra mujeres e inmigrantes con el objetivo de servir para reducir el odio en redes sociales en un futuro y participar en la competición de la Tarea 5 de SemEval-2019.

La Tarea 5 de SemEval-2019 consiste en la detección de discurso de odio contra mujeres e inmigrantes en Twitter, tanto en inglés como en español. Este trabajo propone una sólida base para la detección de discurso de odio haciendo uso de técnicas tradicionales de Machine Learning. Nuestro sistema se basa principalmente en el uso de n-gramas, análisis de sentimientos y word embeddings en conjunto.

En la competición, una de las tareas se basa en detectar discurso de odio contra mujeres e inmigrantes dado un tweet, nuestro sistema obtuvo la segunda mayor precisión en la Tarea A en español, superando a sistemas más complejos basados en redes neuronales de un total de 40 participantes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This bachelor thesis presents the system developed at University of Alicante focused on multilingual detection of hate speech against immigrants and women in Twitter. The project was led and supervised by professors Jose Garcia-Rodriguez, David Tomás Díaz and PhD student Alberto Garcia-Garcia.

The Introduction is organized in four different sections: Section 1.1 introduces the motivation that has led to develop this project, Section 1.2 presents the main and specific goals of the project, along with the planning. Finally, Section 1.3 details the structure of this document.

1.1 Motivation

The motivation of this work is multiple. First, to fight against the existing sexism and racism in social networks. Cyberbullying is a digital era problem, and not enough is being done to prevent it. Due to the massive rise of users in social media, the presence of verbal abuse, hate speech and bully-attitudes has increased over the years. Especially on

Twitter, users find a way to anonymously harass and offend other individuals or collectives, and what's worse, people affected by cyberbullying also often suffer harassment in real life. One of the easiest targets of suffering cyberbullying, apart from teenagers, are women and immigrants, that's the reason in this work we propose a tool that can help the police to protect the victims and can serve governments or technology companies to fight against cybercriminals.

The second motivation is the participation in a SemEval-2019 challenge. SemEval is an International Workshop in Semantic Evaluation sponsored by SIGLEX and Microsoft. Specifically, we have participated in SemEval-2019 Task 5: *HatEval: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter*. More information about the conditions, deadlines and results of the challenge are detailed in Section 4.1.

Finally, the opportunity of diving into the Natural Language Processing world and develop our professional career.

1.2 Goals

The main goal of this project is the development of a well-performing system able to detect hate speech against immigrants and women.

Regardless of the result, the purpose of this work is to build a strong baseline for hate speech detection to be presented to the SemEval challenge, using a traditional machine learning approach with standard textual features, which could serve as a reference to compare with deep learning systems. Deep Learning techniques are currently being applied to numerous Machine Learning tasks, obtaining state-of-the-art results in many of the tasks it is applied to. This project is a first approach to the traditional Machine Learning techniques, and it will act as a reference for our own future work. An extensive analysis on the Natural Language Processing traditional methods will be carried out

and as a secondary goal, we will compare their performance with the Deep Learning approaches presented to the challenge.

The planning of the project, tasks to do and challenge important dates are shown in the Figure 1.1

Month	Task	Challenge
September	Document: Introduction	Training and development data ready
October	State of the art research	
	Document: State of the art	
November	System development begins	
	Datasets reading	
	Bag-of-n-grams	
December	Sentiment analysis	
	Parameter tuning	
	Feature selection	
	Testing	
	Document: Methodology	
January	Count of plurals and insults	
	Testing	Challenge evaluation
Febraury	Word embeddings	Results are notified to participants
	System description paper writing	System description paper submission due
March	Document: Hate Speech Detection System	Paper reviews due
		Author notifications
April	Document: Evaluation	Camera ready submission due
May	Document: Conclusions	
	Final review	
June		SemEval-2019 congress in Minneapolis, USA

Figure 1.1: Project's planning.

1.3 Structure

This document is structured as follows: Chapter 1 outlined the project and presented a detailed state of the art of the issues related to this work. It exposed the motivation of the work and its general and specific goals. Chapter 2 elaborates a state of the art of text classification, and particularly of hate speech detection. Chapter 3 exposes the methodology followed during the development of this project. It presents the different

technologies and tools used to carry out this work. Chapter 4 explains the SemEval-2019 challenge and the hate speech detection system developed in this work. Chapter 5 presents the experiments, results and the discussion for each part of the implementation. Finally, Chapter 6 details the conclusions extracted from this projects and draws future work and directions.

Chapter 2

Related work

Humans have been writing things down for thousands of years, and over the time, our brain has gained an immense amount of experience understanding natural language. However, computers don't have yet the same intuitive understanding of natural language that humans do. Natural Language Processing (NLP) is a branch of Artificial Intelligence that is focused on enabling computers to understand and process human languages, to get computers closer to a human-level understanding of language. The purpose of the SemEval challenge is exactly this: to enable computers to "read" a tweet and detect hate speech.

Along with the interest in NLP, the automated categorization of texts into predefined categories has witnessed a booming interest in the last twenty years, due to the increased availability of documents in digital form and the ensuing need to organize them [2].

In classification-related tasks, one of the most interesting aspects distinguishing different approaches is which features are used. In the following list, we will describe the most relevant set of features used in text categorization, explained more in detail in Schmidt survey [3]:

1. **Simple Surface Features**. The most obvious information to utilize are surface-level features, such as bag-of-n-grams. A bag-of-n-grams model records the number of times each n-gram appears in each document of a collection. A n-gram is a collection of n successive words. Bag-of-n-grams are often reported to be highly predictive, and can be easily combined with other sets of features to improve performance. Indeed, bag of n-grams are used by a majority of authors [4]. Bag-of-n-grams features are discussed in detail in Section 4.4.1.1.

2. **Sentiment analysis**. There are several text classification tasks where sentiment analysis is crucial. For instance, in hate speech detection, since it is safe to assume that usually negative sentiment pertains to a hate speech message. This is the reason why several approaches incorporate sentiment analysis as an auxiliary feature for text classification [5]. Sentiment analysis features are discussed in detail in Section 4.4.1.2.

3. **Word embeddings**. Features such as bag-of-n-grams require to appear in both training and test data to be effective. However, since there are text classification tasks applied on small pieces of text (passages or even individual sentences), one may face a data sparsity problem. In order to solve this problem, distributed word representations, also refered to as word embeddings have been proposed. Word embeddings are explained in detail in Section 4.4.1.3.

4. **Linguistic Features**. Linguistic aspects also play an important role for text classification. Although there are tasks in which linguistic features don't play an important role, they are vital in others. For instance, if the proposed task is to identify if hate speech is towards an individual or a generic group, identifying linguistic features such as plurals may be very relevant. Indeed, in [6] linguistic features as a combination with bag-of-n-grams are explored.

5. **Meta-Information**. If a task involves social networks, meta-information such as

information about the user of a post or tweet may be very predictive. For instance, a user who is known to tweet hate speech tweets may do so again, and a user who is not known to write such messages is unlikely to do so in future. This heuristic is effectively employed in inferring further hate speech messages in [7].

6. **Multimodal Information**. Modern social media do not only consist of text messages but also include images, video and audio content. This context outside a written user comment can be used as a predictive feature. Not too many contributions exploit this type of information, and this is slightly surprising since visual context plays a major role in several tasks.

In the following section, we will discuss more in detail the problem within text categorization on which this work is based: hate speech detection. Finally, although in the research community the dominant approach to text categorization problems is based on machine learning techniques, we will study the recent deep learning approaches for text categorization.

2.1 Hate Speech Detection

Online hate speech is spreading widely, forming a serious problem that can lead to actual hate crimes [8]. Many countries adopted laws prohibiting hate speech where people convicted of using hate speech can face large fines and even imprisionment. So far, to assure there is no spread of illegal hate speech, the European Union has created a code of conduct for social media platforms [9] that needs to be followed. However, these laws are not entirely effective.

Although platforms forbid hateful speech its detection is still a challenging task due to a number of reasons. Some of the issues for detecting hate speech are discussed in [10]. First, tweets are short and full of grammar and syntactic flaws, which makes harder to

use natural language processing tools to extract text-based attributes. Moreover, hate speech can cross sentence boundaries and be present in sarcastic comments in the same voice as the people that were producing abusive languages. Furthermore, hate speech content tends to be ambiguous and context-dependant [11]. Finally, Twitter is full of spam accounts [12], often using vulgar language and exhibiting behaviour that could also be considered as offensive or aggressive. Filtering out such accounts from actual abusive users may be a difficult task.

In order to deal with these issues, over the past few years several techniques to detect hate speech and abusive language online have been proposed. Although related, it is important to distinguish between hate speech and abusive or offensive language. While the former is used to express hatred towards a targeted group based on characteristics such as race, ethnicity, gender and sexual orientation, the latter can be used in the usual language of some users without being hateful [13]. Previous works made use of different machine learning and deep learning techniques that were shown to be valid, such as Support Vector Machines (SVMs) [14], Recurrent and Convolutional Neural Networks (RNNs and CNNs) [15] as well as Long Short Term Memory models (LSTMs) [16]. Regarding the features, prior works made use of heterogeneous features such as bag of words, n-grams, punctuation, as well as lexical features and user-related features [11]. In addition to these features, previous approaches showed the effectiveness of using word embeddings to detect abusive language in social media [17] and exposed how sentiment analysis can also contribute to hate speech and offensive language detection [18].

2.2 Deep Learning Approaches

For a long time, the majority of methods used to study NLP problems employed machine learning models. However, with the recent popularity and success of word embeddings (low dimensional, distributed representations), neural-based models have achieved supe-

rior results results on various language-related tasks as compared to traditional machine learning models like SVM or logistic regression.

In the following subsections, some of the most relevant neural networks for text classification will be discussed.

2.2.1 Recurrent Neural Network

Recurrent neural networks (RNNs) were initially created in the 1980's, but can only show their real potential since a few years ago, because of the increase in available computational power. RNNs are a powerful model for sequential data [19]. They are especially powerful in use cases in which context is critical to predicting an outcome and are distinct from other types of artificial networks because they use feedback loops to process a sequence of data that informs the final output, which can also be a sequence of data. RNNs have received the most success when working with NLP due to their context dependancy as they work with words and paragraphs [20] [21].

While in principle the recurrent neural network is a simple and powerful model, in practice, it is hard to train properly [22]. Among the main reasons why this model is so unhandy are the vanishing gradient and exploding gradient problems described in [23].

2.2.2 Hierarchical Attention Network

In [24], Yang introduced a new neural architecture for document classification, the Hierarchical Attention Network (HAN). Since documents have a hierarchical arquitecture (words form sentences, sentences form documents), the intent is to derive sentence meaning from the words and then derive the meaning of the document from those sentences. Therefore, an attention model is built to pay more attention to the important words.

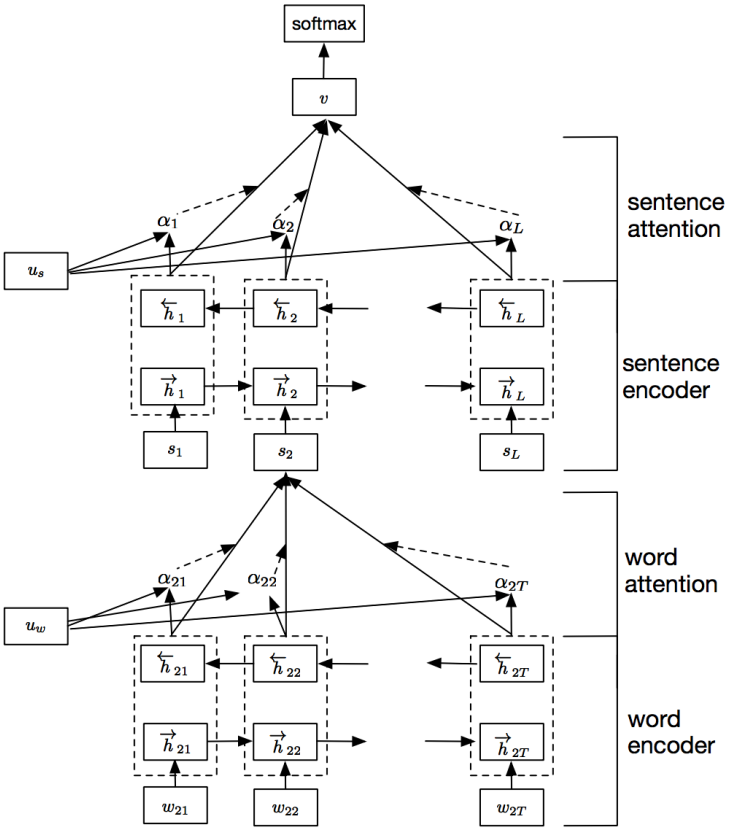


Figure 2.1: Hierarchical Attention Network.

2.2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) is a neural network that can make use of the internal structure of data such as the 2D structure of image data through convolution layers, where each computation unit responds to a small region of input data [25]. CNN has been very succesful in image classification and computer vision; see e.g., the winning solutions of ImageNet Large Scale Visual Recognition Challenge [26] [27]. On text, it has been shown that CNN can be directly applied to distributed [28] or discrete [25] embeddings of words, without any knowledge on the syntactic or semantic structures [29].

Chapter 3

Methodology

In order to face this project, we will need a set of technologies and tools that will help us to carry it out. This chapter presents the technologies and tools employed in the development of this project: in Section 3.1, and in Section 3.2 we describe the datasets used in the evaluation of the project.

3.1 Technologies

In the following subsections we will describe the different technologies and tools which were directly or indirectly used during the development of this project.

3.1.1 Python

Python is an interpreted, object-oriented, high-level programming language. Its extensive collection of libraries and packages available ease the development of applications and its simple, easy to learn syntax therefore reduces the cost of program maintenance.

Its simplicity and its particular tools, extensions and frameworks lead Python to the top of the programming languages for machine learning.

3.1.2 spaCy

spaCy[1] is an open-source software library for advanced Natural Language Processing, written in the programming languages Python and Cython. It is easy to install, and provides a simple and productive API. It is idoneous to prepare text for machine learning, and linguistically sophisticated statistical models can be easily constructed.

The central data structures in spaCy are the *Doc* and the *Vocab*. The *Doc* object owns the sequence of tokens and all their annotations. The *Vocab* object owns a set of look-up tables that make common information available across documents.

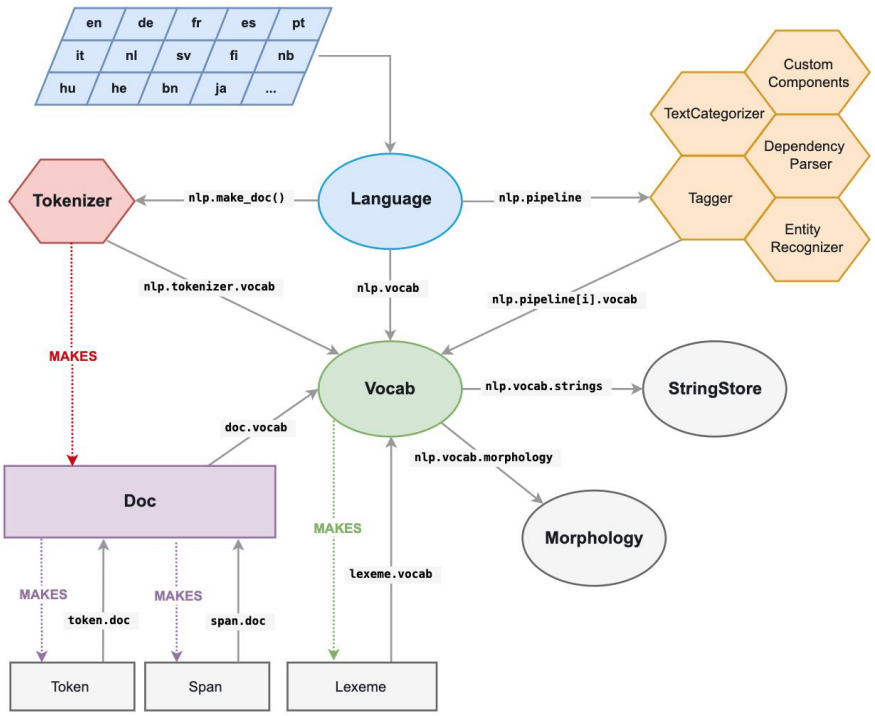


Figure 3.1: spaCy library architecture.

[1] https://spacy.io/

3.1.3 Scikit-learn

Scikit-learn[2] [30] is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN. Scikit-learn answers the growing need for statistical data analysis by non-specialists in the software and web industries, as well as in fields outside of computer-science, while maintaining an easy-to-use interface tightly integrated with the Python language.

3.1.4 NumPy

NumPy[3] is the fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy arrays are the standard representation for numerical data, enabling efficient implementation of numerical computations in a high-level language [31]. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Also, arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

3.1.5 Natural Language Toolkit

The Natural Language Toolkit, NLTK[4], is a suite of open source program modules, tutorials and problem sets, providing ready-to-use computational linguistics courseware [33]. NLTK covers symbolic and statistical natural language processing, providing classification, tokenization, stemming, tagging, parsing, semantic reasoning and an active

[2] `https://scikit-learn.org/stable/`
[3] `www.numpy.org`
[4] `www.nltk.org`

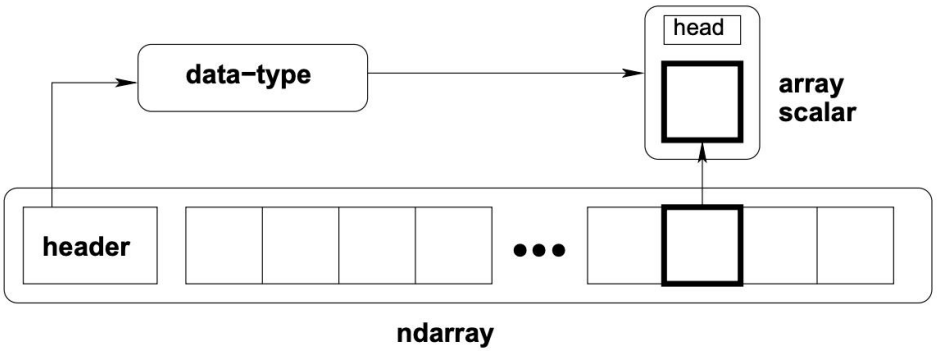


Figure 3.2: Conceptual diagram showing the relationship between the three fundamental objects used to describe the data in an array: 1) the ndarray itself, 2) the data-type object that describes the layout of a single fixed-size element of the array, 3) the array-scalar Python object that is returned when a single element of the array is accessed [32].

discussion forum. A wide guide for its use is available in [34].

3.1.6 TextBlob

TextBlob[5] [35] is a Python library for processing textual data. It provides a simple and consistent API for diving into common Natural Language Processing tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

3.2 Datasets

In this project's experiments, we used the two datasets made available as part of the SemEval-2019 Task 5 challenge: one for English and another for Spanish. The datasets consist of annotated hate speech tweets, equally distributed between hate speech to women and immigrants, and are composed of training, development and test splits. We

[5] `https://textblob.readthedocs.io/en/dev/`

made use of the training split to train our system, and we validated it with the development split, getting ready for the final evaluation: the test split. Also, the tweets are annotated on target classification and aggressive behaviour. For English, the number of tweets for each split is 9100, 1000 and 2971 for training, development and test, respectively. Conversely, the Spanish splits contain 4600, 500 and 1600 tweets.

Tables 3.1 and 3.2 show the distribution of the datasets. This information has been extracted from the SemEval-2019 Task 5 official paper [36].

Label	**Training**		**Test**	
	Imm.	**Women**	**Imm.**	**Women**
Hateful	39.76	44.44	42.00	42.00
Non-Hateful	60.24	55.56	58.00	58.00
Individual Taget	5.89	64.94	3.33	80.63
Generic Target	94.11	35.06	96.67	19.37
Aggressive	55.08	30.06	59.84	34.44
Non-Aggressive	44.92	60.94	40.16	65.56

Tabla 3.1: Distribution percentages across sets and categories for English data. The percentages for the target and aggressiveness categories are computed on the total number of hateful tweets.

Label	**Training**		**Test**	
	Imm.	**Women**	**Imm.**	**Women**
Hateful	41.93	41.38	40.50	42.00
Non-Hateful	58.07	58.62	59.50	58.00
Individual Taget	13.72	87.58	32.10	94.94
Generic Target	86.28	12.42	67.90	5.06
Aggressive	68.58	87.58	50.31	92.56
Non-Aggressive	31.42	12.42	46.69	7.44

Tabla 3.2: Distribution percentages across sets and categories for Spanish data. The percentages for the target and aggressiveness categories are computed on the total number of hateful tweets.

Chapter 4

Hate Speech Detection System

In this section we describe our hate speech detection model and the SemEval 2019 challenge. The main goal of our model is to identify hate speech given a piece of text, in this case a tweet. The specifications of the SemEval 2019 challenge are detailed in Sections 4.1 and 4.2. A high-level overview of our model is presented in Section 4.3 and the set of features that are used in our model are described in Section 4.4.

4.1 SemEval 2019 Challenge

SemEval (**Sem**antic **Eval**uation) is an ongoing series of evaluations of computational semantic analysis systems. SemEval-2019 [1] announced 12 tasks. Specifically, we participated in *Task 5: Hateval: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter* [2] [37] and we based almost the entire work in the task.

The competition was carried out in CodaLab. CodaLab Competitions is a powerful

[1] `http://alt.qcri.org/semeval2019/`
[2] `https://competitions.codalab.org/competitions/19935#learn_the_details`

source framework for running competitions that involve result or code submission.

Participants are encouraged to submit a 4-page system description paper describing their systems and submissions [3] and to assist to SemEval annual conference. This year SemEval-2019 will be held June 6-7, in Minneapolis, USA, collocated with NAACL-HLT 2019. Best-performing participants of each task are allowed to present their work in the conference.

4.2 Task 5: HateEval: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter

The proposed task consists in hate speech detection in Twitter but featured by two specific different targets, immigrants and women, in a multilingual perspective, for Spanish and English. Indeed, race and gender hate speech has become an increasingly important issue in social media, as it stands for 50% of the targets of hate speech in Twitter [38].

The task was articulated around two related subtasks for each of the involved languages: a basic task about hate speech (Task A), and another one where fine-grained features of hateful contents were investigated in order to understand how existing approaches may deal with the identification of especially dangerous forms of hate, i.e. those where the incitement is against an individual rather than against a group of people, and where an aggressive behavior of the author can be identified as a prominent feature of the expression of hate (Task B). Participants were asked to identify, on the one hand, if the target of hate is a single human or a group of persons. On the other hand, if the message author intends to be aggressive, harmful, or even to incite, in various forms, to violent acts against the target.

[3] We submitted our paper describing our model to the task and it was accepted. It is available in `https://github.com/CPerelloC/UA-SemEval`

4.2.1 Task A - Hate Speech Detection against Immigrants and Women

Task A proposes a two-class (or binary) classification where systems have to predict whether a tweet in English or Spanish with a given target (women or immigrants) is hateful or not hateful (HS).

Evaluation

Systems were evaluated using standard evaluation metrics, including accuracy, precision, recall and F1-score. The submissions were ranked by F1-score. The metrics were computed as follows:

- **Accuracy**: *(number of correctly predicted instances) / (total number of instances)*
- **Precision**: *(number of correctly predicted instances) / (number of predicted labels)*
- **Recall**: *(number of correctly predicted labels) / (number of labels in gold standard)*
- **F1-score**: $(2 * Precision * Recall)/(Precision * Recall)$

4.2.2 Task B - Aggressive Behavior and Target Classification

Task B participants were asked first to classify hateful tweets for English and Spanish (e.g., tweets where hate speech against women or immigrants has been identified) as aggressive or not aggressive (AG), and second to identify the target harassed (TR) as individual or generic (i.e. single human or group).

Now that both tasks are introduced, regarding the format of the datasets, they are structured as follows:

```
id      text    HS      TR      AG
21256   Let's build that wall.#BuildThatWall 1   0   1
21299   This immigrant should be hung or shot!    1   1   1
21565   And you are the immigrant   0   0   0
```

Evaluation

Systems will be evaluated on the basis of two criteria: partial match and exact match.

- **Partial match**: each dimension to be predicted (Hate Speech, Target Classification and Aggressive Behaviour) will be evaluated independently of the others using standard metrics, including *accuracy*, *precision*, *recall* and *F1-score*.

- **Exact match**: all the dimensions to be predicted will be jointly considered computing the exact match ratio (EMR) [39]. The EMR scores measures the percentage of instances which are correctly labeled in all subtasks, i.e., hate speech, target classification and aggressive behaviour.

The submissions will be ranked by EMR.

4.3 Approach

Our model consists of a linear classifier based on Support Vector Machines [4] (SVMs) [40]. We have based our model on SVMs since they have proved to provide competitive results in text categorization since their conception [41].

SVM is a supervised machine learning algorithm which can be used for classification

[4] `https://scikit-learn.org/stable/modules/svm.html`

or regression problems. It is a prediction tool that uses a technique called the kernel trick to transform the data and then based on these transformations it finds an optimal boundary between the possible outputs. The foundations of SVMs have been developed by Vapnik [42] and gained popularity due to many promising features such as better empirical performance. SVMs were first developed to solve the classification problem, but recently they have been extended to solve regression problems [43].

The advantages of SVMs are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels. Some of the main SVM kernels are described in Section 5.1.4.

Nevertheless, SVMs include the following disadvantages:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

The operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance (margin) to the training examples. Therefore, the optimal separating hyperplane maximizes the margin of the training data. In Figure 4.1 the

task to find the optimal hyperplane is simple as the data space is linearly separable. Nevertheless, difficulties arise when the data space is not linearly separable.

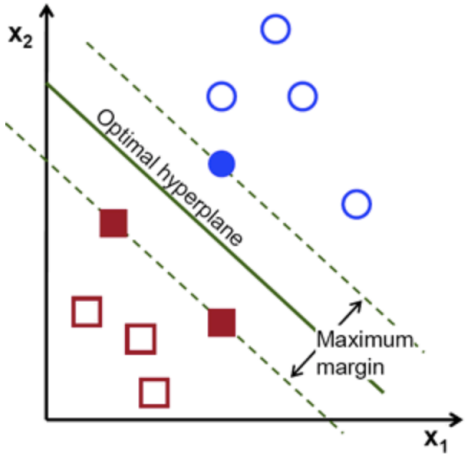


Figure 4.1: SVM's optimal hyperplane in a linearly separable data space.

Consider the Figure 4.2 example where the goal is to find a hyperplane that separates the two classes. Although the space is non-linearly separable, SVM can solve this problem with the kernel trick. In this particular example, the transformation takes place by introducing a new feature $z = x^2 + y^2$ and plotting the z feature.

(a) Non-linearly separable problem (b) Linearly separable problem

Figure 4.2: Conversion of a non-linearly separable problem to a linearly separable problem with the kernel trick.

Our SVM classifier is trained on tweets containing hate speech annotations. During

training, the model is fed with features relevant to hate speech detection. Then, in the test phase the goal of our model is to classify unannotated tweets with the categories learned during the training phase. In the following section we describe the main features used in our SVM classifier.

4.4 Features

In the following we describe the features used for both Task A and Task B of the SemEval challenge. Although the feature configurations for both languages, English and Spanish, are similar, they are not identical. While in English we make use of sentiment analysis features, in Spanish we utilise the length of the tweet in words as a feature.

4.4.1 Task A

For the main Task A (hate speech detection) we distinguish three group of features: *bag-of-n-grams, sentiment analysis and word embeddings.* In addition to these three groups, we use an extra standard feature, the *length of the tweet* in words.

4.4.1.1 Bag-of-n-grams

The most commonly used text representation feature is bag-of-words [41]. Bag-of-words is a simple algorithm used in NLP to extract features from text documents. These features can be used for training machine learning algorithms, such as SVMs. It creates a vocabulary of all the unique words occurring in all the documents in the training set. In the bag-of-words, grammar and word order are disregarded. However, compared to bag-of-words, bag-of-n-grams considers not only words, but also consecutive words (n-grams) and they mind the words order.

Bag-of-n-grams features, which have been already used for hate speech detection [11], are often reported to be highly predictive and can be combined with other features to improve performance. We make use of unigrams, bigrams and trigrams, represented in the feature vectors by their frequency in a tweet. Unigrams handle words individually, bigrams take two consecutive terms, and trigrams three.

Libraries such as NLTK provide methods to create an n-gram language model to capture patterns in n consecutive words of training text. Instead, we have created our own methods to extract n-grams in tweets to have more control over specific parameters.

The following example will help understand more deeply the algorithm to extract and treat the bag-of-n-grams. For its simplicity, in the example only unigrams and bigrams are taken into account, although we also manage trigrams in our algorithm. Consider the below two tweets.

```
1. "Peter likes to go to the beach."
2. "Peter also likes the mountains."
```

These two sentences can be also represented with a collection of words, known as bag-of-words.

```
1. ['Peter', 'likes', 'to', 'go', 'to', 'the', 'beach']
2. ['Peter', 'also', 'likes', 'the', 'mountains']
```

Once the bag-of-words has been formed, n-grams are added to form the bag-of-n-grams.

```
1. ['Peter', 'likes', 'to', 'go', 'to', 'the', 'beach', ('Peter',
 'likes'), ('likes', 'to'), ('to', 'go'), ('go', 'to'), ('to', 'the'
 ), ('the', 'beach')]
2. ['Peter', 'also', 'likes', 'the', 'mountains', ('Peter', 'also'
```

```
), ('also', 'likes'), ('likes', 'the'), ('the', 'mountains')]
```

Further, for each tweet, the multiple occurrences of the n-grams are removed and the word count is used to represent this.

```
1. {'Peter':1, 'likes':1, 'to':2, 'go':1, 'the':1, 'beach':1,
('Peter', 'likes'):1, ('likes', 'to'):1, ('to', 'go'):1, ('go',
 'to'):1, ('to', 'the'):1, ('the', 'beach'):1}
2. {'Peter':1, 'also':1, 'likes':1, 'the':1, 'mountains':1,
('Peter', 'also'):1, ('also', 'likes'):1, ('likes', 'the'):1,
('the', 'mountains'):1}
```

Assuming these tweets are part of the same training dataset, the n-gram frequency of the entire dataset is combined.

```
{'Peter':2, 'likes':2, 'to':2, 'go':1, 'the':2, 'beach':1,
('Peter', 'likes'):1, ('likes', 'to'):1, ('to', 'go'):1, ('go',
'to'):1, ('to', 'the'):1, ('the', 'beach'):1, 'also':1,
'mountains':1, ('Peter', 'also'):1, ('also', 'likes'):1,
('likes', 'the'):1,('the', 'mountains'):1}
```

The above vocabulary from all the tweets in the training dataset, with their respective n-gram frequency, will be used to create the vectors for each of the tweets. In order to represent our original tweets in a vector, each vector is initialized with all zeros. Next, for each n-gram of the vocabulary, its value in the vector will be incremented if the tweet has that n-gram. Finally, the vector representations of the tweets are obtained and the SVM is ready to use.

```
1. "Peter likes to go to the beach."
[1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
2. "Peter also likes the mountains."
```

```
[1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]
```

4.4.1.2 Sentiment analysis

Sentiment analysis is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes [20].

Not surprisingly, the most important indicators of sentiments are sentiment words. These are words that are commonly used to express positive or negative sentiments. For example, *good, wonderful* and *amazing* are positive sentiment words, and *bad, poor* and *terrible* are negative sentiment words. Apart from individual words, phrases are also essential to sentiment analysis.

Although sentiment words and phrases are important for sentiment analysis, only using them is far from sufficient. The problem is much more complex. Below, we highlight several issues, explained more in detail in [20]:

1. A positive or negative sentiment word may have opposite orientations in different application domains. For example, "suck" usually indicates a negative sentiment, e.g., *"This laptop sucks"*, but it can also imply a positive sentiment, e.g., *"This tumble dryer really sucks"*.

2. A sentence containing sentiment words may not express any sentiment. This is common in questions and conditionals. For example, *"Why does your laptop suck?"*.

3. Sarcastic sentences with or without sentiment words are hard to deal with, e.g., *"What a fantastic laptop! It stopped working in two days"*.

4. Many sentences without sentiment words can also imply opinions. As an example, *"After using this laptop for two days, a key has slipped out"*.

Hate speech and *sentiment analysis* are closely related, and we can assume that negative sentiment usually pertains to a hate speech message [3]. To integrate this feature into our model we simply add the output of a pretrained sentiment analysis classifier in the tweets' vector representation.

4.4.1.3 Word embeddings

Word embeddings are dense vector representations of wordforms in which similar words are expected to be close in the vector space [5] Word embeddings are used for semantic parsing, to extract meaning from text to enable natural language understanding, and for a language model to be able to predict the meaning of a text, it needs to be aware of the contextual similarity of words. Word embeddings are used extensively in natural language processing [44].

For example, Figure 4.3 represents an animal word space. In the figure, all the big cats (i.e. cheetah, jaguar, panther, tiger and leopard) are really close in the vector space. Word embeddings represent one of the most successful applications of unsupervised learning, mainly due to their generalization power. The construction of these word embeddings varies, but in general a neural language model is trained on a large corpus and the output of the network is used to learn word vectors (e.g. Word2Vec [45]). A recent survey[6] in this topic analyzes more in depth how to build word embeddings.

Moreover, in [46] Bayot and Gonçalves showed that word embeddings provide a useful generalization signal in text classification when used in a similar setting. In our case, we add the average of the word embeddings in a tweet as an additional set of features

[5] We would recommend the reading of Jose Camacho Collados' Medium post on the contribution of neural networks and word embeddings in Natural Language Processing.

[6] `http://ruder.io/word-embeddings-2017/index.html#evaluation`

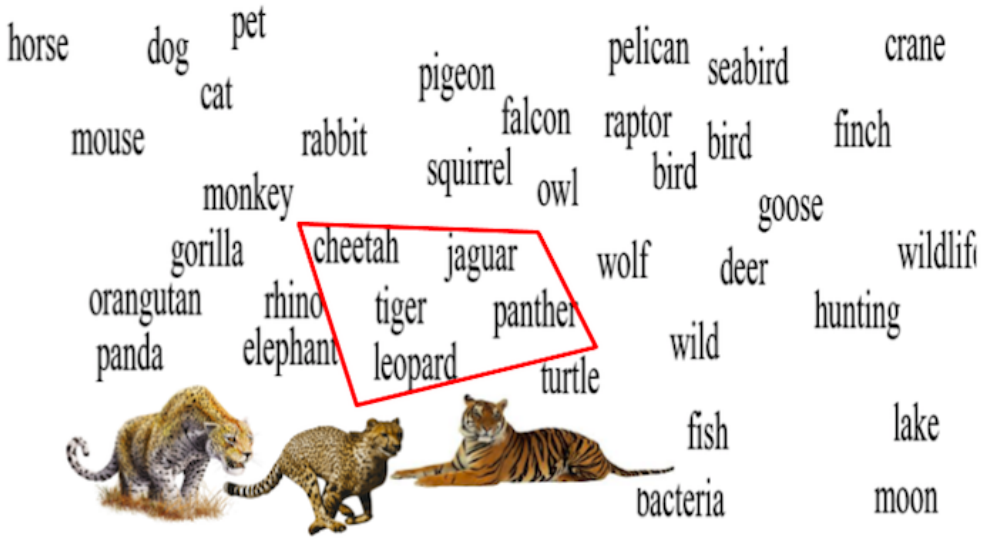


Figure 4.3: Cluster of the Word2Vec vector space reduced to two dimensions using t-SNE [1]

in our SVM classifier. We made use of Spanish and English 100-dimensional FastText word embeddings [47] trained on two large Twitter corpus from Spain and United States, respectively [48].

4.4.2 Task B

For Task B, we use two simple extra features with specific information about each subtask.

4.4.2.1 Count of the plural nouns

For *target classification* (individual or collective), we use the count of the plural nouns in the tweet as a feature. This feature is very indicative of whether the hate speech is towards an individual or towards a collective. As an example, if the word *"immigrants"* appears in a hate speech tweet, in most cases, hate speech is towards the immigrant

community.

In order to do this, we use spaCy *part-of-speech* (PoS) tagger[7]. PoS tagging is the task of labeling words as one of several categories to identify the word's function in a given language, i.e. if a word is an adjective, a noun, a verb, etc. In case the word is a noun, it allows to identify if it is singular or plural. PoS tagging itself may not be the solution to any particular NLP problem, nevertheless, provides very useful information to other NLP tasks, such as information extraction and name entity recognition [49], or in this case, text categorization. PoS tagging is not a trivial task, and moreover, Twitter poses additional challenges due to the ambiguous nature of the text and the lack of conventional ortography [50].

Statistical models enable spaCy's PoS tagger to make a prediction of which tag or label most likely applies in the context of a sentence.

4.4.2.2 Count of the insults in the tweet

For *aggressive behaviour*, we use the count of the insults in the tweet as a feature. We hypothesize that a high level of insults may involve violent behaviour. To this end, we filter a database from insults collected at `https://hatebase.org/`.

Hatebase is the world's largest structured repository of regionalized, multilingual hate speech. It storages 2960 terms in 97 languages classified for their severity. However, in Spanish this classification is not accurate, and we decided to build our own filter of insults based on the database.

[7] `https://spacy.io/usage/linguistic-features`

Chapter 5

Evaluation

In this section we describe the experimental setup (Section 5.1) of our system along with the results obtained (Section 5.2), including a brief analysis of errors detected in the evaluation phase of the challenge (Section 5.3).

5.1 Experimental setup

In the following we present details about the text preprocessing (Section 5.1.1) and feature selection procedures (Section 5.1.2), the pre-trained models used as part of our model (Section 5.1.3) and how parameter tuning was performed (Section 5.1.4).

5.1.1 Preprocessing

Each tweet is tokenized using the spaCy NLP library. We experimented with various preprocessing variants and decided to work with raw words as tokens (i.e., without applying lemmatization) because the lemmatizer didn't work good enough with the casual Twit-

ter language, removing punctuation and URLs but keeping emojis and stopwords since pronouns and articles, especially on Spanish that articles have genre, can be relevant in the context of hate speech classification.

5.1.2 Feature selection

One of the main issues in text classification is the high dimensionality of the feature space [51]. For instance, over 300,000 and 150,000 features were initially obtained using the bag-of-n-grams features alone on, respectively, the English and Spanish training sets from Task A. Besides the computational cost of training a model with such a large amount of features, an additional issue is the noise that could be introduced by including many irrelevant features. Thus, it is generally desirable to reduce the feature space, without sacrificing classification accuracy.

Some of the most important feature selection methods are the tree-based feature selection, in which tree-based estimators are used to compute feature importances, which in turn can be used to discard irrelevant features, and the mutual information method used for feature selection, measuring the dependency between the features. Nevertheless, the feature selection method used in our system is based on word frequency, understanding a word as an n-gram. The system first delimits the n-grams by a frequency number to significantly reduce the feature space before preparing the vectors for the SVM. Then, highly sparse features (i.e. containing zero in more than 99.9% of the samples) are removed by leveraging the *VarianceThreshold* tool from *scikit-learn* [1][30].

After the feature selection is performed for the bag-of-n-grams features, the featured space is reduced from 336,669 to 4,605 in English Task A and from 177,003 to 4,217 in Spanish Task A.

[1] `https://scikit-learn.org/stable/modules/feature_selection.html`

5.1.3 Pre-trained models

Regarding sentiment analysis, we used as features the polarity [-1.0, 1.0] and the subjectivity [0.0, 1.0] of a tweet according to TextBlob [2] [35]. Note that TextBlob is only optimized for English input and was not used for the Spanish tasks.

As we described in Section 4.4.1.3, as far as word embeddings are concerned, we made use of Spanish and English 100-dimensional FastText word embeddings [47] trained on two large Twitter corpus from Spain and United States, respectively [48].

5.1.4 Parameter tuning

We experimented with several kernels and parameter configurations to train the Support Vector Machines, including linear and radial basis function (rbf) kernels. In an early state of the project, we tested which kernel would fit better and obtain better results with a basic configuration. With Task A Spanish development set as validation, we tested three kernels: the linear kernel, the rbf kernel and the polynomial kernel. We used SVM's default configurations except for the *gamma* parameter and the bag-of-n-grams as only features.

Kernel	**Gamma**			
	Auto		**Scale**	
	Acc	F1-score	Acc	F1-score
Linear	**72.7**	**72.0**	**72.7**	**72.0**
Rbf	62.7	53.8	68.3	65.1
Poly	55.5	35.7	55.5	35.7

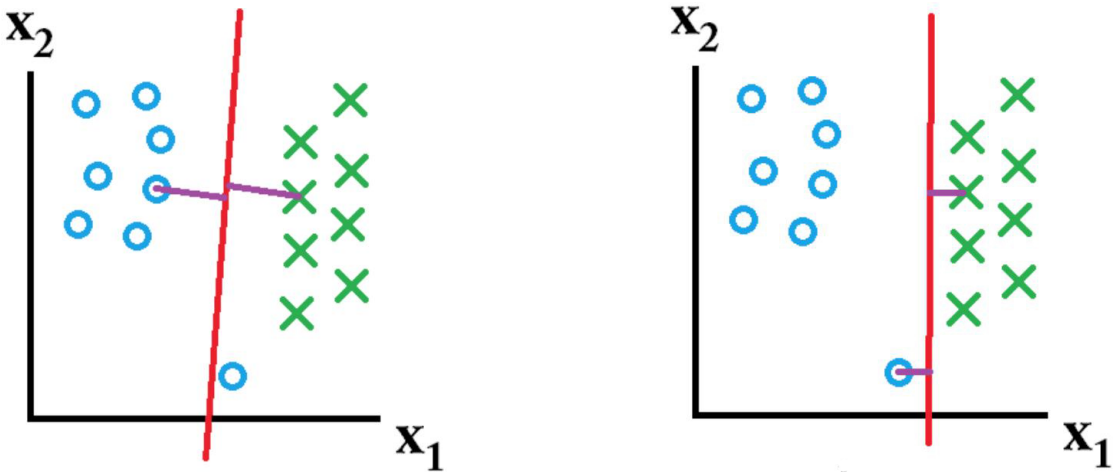
Tabla 5.1: Task A results with different kernel configurations using bag-of-n-grams as features.

Table 5.1 shows the linear kernel is the best performing kernel with a basic configuration. Since our system is trained with a large amount of features, the linear kernel

[2] `https://textblob.readthedocs.io/en/dev/index.html`

fits perfectly. That's because mapping the data to a higher dimensional space does not really improve the performance [52]. Also, it is hard to find an optimal parameter configuration for the polynomial and the rbf kernel. Therefore, we decided to use a linear kernel, as the SVM training was faster, implied tuning less parameters and is indeed very well suited for text categorization.

To train a SVM with a linear kernel, it is imperative to optimize the C regularization parameter. The SVM training seeks two things: a hyperplane with the largest minimum margin and a hyperplane that correctly separates as many instances as possible. The problem is, accomplishing both is not an easy task. The C parameter is the responsible for regulating this hyperplane.



(a) Low C (b) Large C

Figure 5.1: C parameter influence in SVMs training.

Figure 5.1 shows an illustrated example of C parameter's influence in the SVM training process. We fine-tuned the C parameter of the SVM using as validation the development set of the task. This parameter tuning was performed using bag-of-n-grams as features and on the Spanish dataset only, serving as a reference for the English dataset too.

Table 5.2 shows the results for the different C parameter configurations. The value of C that achieved the highest accuracy in the development set was $C = 2^{-5}$ for Task

C parameter	1	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
Accuracy	73.6	75.0	77.0	76.4	**77.8**	74.2

Tabla 5.2: Task A results with different C parameter configurations using bag-of-n-grams as features.

A and Task B-target classification, and $C = 3$ for Task B-aggressive behaviour, which were fixed across all experiments.

5.2 Results

In the following we present our results for Task A (Section 5.2.1) and Task B (Section 5.2.2).

5.2.1 Task A

Task A consists of detecting hate speech (HS) against women or immigrants in the text. Systems were evaluated according to standard classification metrics such as accuracy and macro-F1-score.

Features	English				Spanish			
	Dev		**T**est		**D**ev		**T**est	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
All	**72.8**	**72.0**	50.1	48.1	**78.4**	**77.9**	**7**3.1	**72.2**
N-grams	72.1	71.5	50.0	48.0	77.2	76.6	73.0	71.9
N-grams, sent. analysis	72.5	71.8	50.1	48.0	-	-	-	-
N-grams, tw. length	-	-	-	-	77.4	76.8	73.0	71.9
Word embeddings	65.3	60.1	**57.5**	**56.9**	63.6	56.3	65.9	55.0
SVC baseline	-	-	*49.2*	*45.1*	-	-	*70.5*	*70.1*
MFC baseline	-	-	*57.9*	*36.7*	-	-	*58.8*	*37.0*

Tabla 5.3: Task A results using different sets of features. The proposals highlighted in bold were submitted to the task.

Table 5.3 shows our Task A results in the development and evaluation sets comparing

different sets of features described in Section 4.4. As can be observed in the table, the highest accuracy and macro-F1-score obtained in the development phase were, respectively, 78.4 and 77.9 using all features (i.e., n-grams, tweet length and word embeddings) for Spanish and 72.8 and 72.0 with n-grams for English (the same features including sentiment analysis in this case).

The sentiment analysis feature provided a small improvement when combined with n-grams on the English development set, but had a negligible influence on the test set. In general, except for the word embeddings which seem to generalize better, all features performed close to a random baseline in the English test. A further analysis should be required to explain the difference between development and test results, which affected most participating systems. Some possible explanations are discussed in the Analysis section (Section 5.3).

Unlike in English, in Spanish our system obtains the best result with the configuration that performed best in the development set. Our official submission (n-grams and tweet length as features) ranked sixth in terms of macro-F1 and second in terms of accuracy among all 40 participating systems. In the English Task, with the addition of word embeddings as feature, our system would have ranked third in terms of macro-F1, among 69 participants.

The complete SemEval-2019 Task 5 ranking is available at `https://docs.google.com/spreadsheets/d/1wSFKh1hvwwQIoY8_XBVkhjxacDmwXFpkshYzLx4bw-0/`.

User name	**Accuracy**	**macro-F1**
hammad.fahim57	75.9	72.7
Pere	**73.6**	**72.5**
luiso.vega	73.4	73.0
francolq2	73.1	73.0
stefanomozart	73.1	72.2
gernert	72.9	72.9

Tabla 5.4: Results of the Top 6 performers in Spanish Task A ordered by Accuracy.

Table 5.4 show the official results for the Spanish Task A Top 6 performers ordered by Accuracy. The results don't match Table 5.3 results because several tweets were removed from the test before evaluating.

5.2.2 Task B

Task B consists of identifying the target harassed as individual or generic (TR), and to classify hateful tweets as aggressive or not aggressive (AG). As previously explained, our official submission consisted of n-grams and sentiment analysis features, with the addition of the two extra features mentioned in Sections 4.4.2.1 and 4.4.2.2: a count of plurals in each tweet for TR and a count of insults for AG.

Table 5.5 displays the results of our system on Task B. As can be observed, results for TR are better for Spanish than English, which could be attributed to the fact that Spanish uses more plural forms than English. Regarding AG, the reason could be that the insult database was more accurately filtered for Spanish than English. These results, however, show the general trend of participating systems in the task.

	F1(HS)	**F1(TR)**	**F1(AG)**	**F1(avg)**	**EMR**
English Dev	71.8	72.7	60.9	68.5	56.9
English Test	48.0	68.2	54.4	56.8	31.2
Spanish Dev	77.9	80.6	81.6	80.0	68.4
Spanish Test	72.2	75.9	73.5	73.9	62.9

Tabla 5.5: Task B results in the development and evaluation phases.

Finally, we noted that training only on the portion of tweets where hate speech is present is beneficial. In our official submission we used all tweets for training, irrespective of whether they were hateful or not. Using all tweets for training was clearly adding a lot of noise to the training, and without it, a significant increase in the performance was obtained.

Table 5.6 shows the results using the full training set and the training set including

Training	English Test		Spanish Test	
	F1(TR)	F1(AG)	F1(TR)	F1(AG)
Full	68.2	54.4	75.9	73.5
Only hateful	**88.0**	**70.9**	**92.8**	**87.8**

Tabla 5.6: Macro-F1 results in Task B by using different types of training data.

tweets considered as hateful. AS an example, in the Spanish test set, the macro-F1-scores using only hateful tweets for trainng were 92.8 and 87.8, which means an absolute improvement of 16.9 and 14.3 percentage points for target classification and aggressive behaviour, respectively.

5.3 Analysis

When analyzing the errors of our system, we found a number of cases where irony was present. It is worth noting that sometimes hate speech is expressed through irony, and therefore does not imply and aggressive behaviour. Moreover, offensive language does not necessarily imply hate speech, which poses an additional challenge to these systems. Here are some sample tweets of hate speech without aggressive behaviour from the development set:

"Say it loud, say it clear, illegal #immigrants are not welcome here."

"Poland: our country is safe because we haven't taken in refugees"

Finally, given the disparity of results between English development and test sets, we analyzed possible causes for this behaviour. In Task A, we obtained the best performance by only using word embeddings. One of the reasons for these results could be that, in the development set 64.8% of the vocabulary of the test set was present in the training set, whereas in the evaluation test only 54.8% of the vocabulary overlapped with the vocabulary of the training set. This reduction in the overlapping vocabulary between

training and test handicaps the performance of n-gram based systems, which heavily relies in vocabulary overlap. Word embeddings are less affected by this condition since they can capture synonymy relations and therefore are able to generalize better. This could explain why using word embeddings alone attained the best performance in this experiment, as the n-grams were not helpful.

Chapter 6

Conclusions

In this bachelor thesis we presented our system on hate speech detection to fight against two crucial problems in social media: sexism and racism. This work may serve governments or other companies to find harassment and bully-attitudes on the net, and can help them to fight against potential criminals.

In this work we described our system presented at SemEval 2019 Task 5. The system follows a traditional machine learning approach based on feature engineering, making use of n-grams, sentiment analysis and word embeddings as its main features. The results obtained show how word embeddings, when combined with n-grams and sentiment analysis, can improve the performance of the system. In Spanish task A, our proposed system obtained a remarkable macro-F1 score of 72.5 (sixth highest) and an accuracy of 73.6 (second highest). In view of these results, we have achieved our objective of building a strong baseline for hate speech detection.

We would like to highlight the acceptation and publication of our paper describing our system in the Association for Computational Linguistics web, leader in Natural Language Processing. The paper is made public in `http://alt.qcri.org/semeval2019/`

data/uploads/semeval2019-proceedings.pdf, page 504, and will be published soon individually.

Future directions of this work include incorporating users' features to the model, studying how the pronouns and the context of the tweet may affect hate speech classification, and comparing the resulting system with deep neural network approaches, which have recently gained popularity in text classification tasks. Also, the testing of the system performance in more datasets, and the posibility of putting into practice its real effectiveness in social networks, not only in Twitter but also Instagram and Facebook.

Bibliography

[1] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[2] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[3] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, 2017.

[4] Sara Sood, Judd Antin, and Elizabeth Churchill. Profanity use in online communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1481–1490. ACM, 2012.

[5] Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230, 2015.

[6] Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North*

American chapter of the association for computational linguistics: Human language technologies, pages 656–666. Association for Computational Linguistics, 2012.

[7] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1980–1984. ACM, 2012.

[8] Mari J Matsuda. Public response to racist speech: Considering the victim's story. In *Words that wound*, pages 17–51. Routledge, 2018.

[9] Věra Jourová. Code of conduct on countering illegal hate speech online: First results on implementation. *European Commission.[cit. 8. březen 2018]*, 2016.

[10] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee, 2016.

[11] Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. Mean birds: Detecting aggression and bullying on twitter. In *Proceedings of the 2017 ACM on web science conference*, pages 13–22. ACM, 2017.

[12] Chao Chen, Jun Zhang, Xiao Chen, Yang Xiang, and Wanlei Zhou. 6 million spam tweets: A large ground truth for timely twitter spam detection. In *2015 IEEE international conference on communications (ICC)*, pages 7065–7070. IEEE, 2015.

[13] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*, 2017.

[14] Shervin Malmasi and Marcos Zampieri. Detecting hate speech in social media. *arXiv preprint arXiv:1712.06427*, 2017.

[15] Dominik Stammbach, Azin Zahraei, Polina Stadnikova, and Dietrich Klakow. Offensive language detection with neural networks for germeval task 2018. In *14th Conference on Natural Language Processing KONVENS 2018*, page 58, 2018.

[16] Julian Risch, Eva Krebs, Alexander Löser, Alexander Riese, and Ralf Krestel. Fine-grained classification of offensive language. In *14th Conference on Natural Language Processing KONVENS 2018*, page 38, 2018.

[17] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30. ACM, 2015.

[18] Vinita Nahar, Sayan Unankard, Xue Li, and Chaoyi Pang. Sentiment analysis for effective detection of cyber bullying. In *Asia-Pacific Web Conference*, pages 767–774. Springer, 2012.

[19] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[20] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[21] Ji Young Lee and Franck Dernoncourt. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827*, 2016.

[22] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages

1310–1318, 2013.

[23] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[24] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[25] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[28] Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[29] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[30] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand

Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[31] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[32] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[33] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[34] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[35] Steven Loria, P Keen, M Honnibal, R Yankovsky, D Karesh, E Dempsey, et al. Textblob: simplified text processing. *Secondary TextBlob: Simplified Text Processing*, 2014.

[36] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019). Association for Computational Linguistics", location="Minneapolis, Minnesota*, 2019.

[37] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019). Association for Computational Linguistics, 2019.

[38] Leandro Araújo Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. Analyzing the targets of hate in online social media. In *ICWSM*, pages 687–690, 2016.

[39] Hideto Kazawa, Tomonori Izumitani, Hirotoshi Taira, and Eisaku Maeda. Maximal margin labeling for multi-topic text categorization. In *Advances in neural information processing systems*, pages 649–656, 2005.

[40] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[41] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[42] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[43] Vladimir Vapnik, Steven E Golowich, and Alex J Smola. Support vector method for function approximation, regression estimation and signal processing. In *Advances in neural information processing systems*, pages 281–287, 1997.

[44] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[45] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[46] Roy Bayot and Teresa Gonçalves. Author profiling using svms and word embedding

averages. In *Proceedeings of the International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. CEUR, 2016.

[47] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146, 2017.

[48] Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 531–535. ACM, 2016.

[49] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[50] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2010.

[51] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420, 1997.

[52] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[53] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[54] Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. What does this emoji mean? a vector space skip-gram model for twitter emojis. In *Language Resources*

and Evaluation conference, LREC, Portoroz, Slovenia, May 2016.

[55] Yiming Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 1995.

[56] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.